UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 10/767,780 | 01/28/2004 | Shingo Fukui | P/1878-188 | 5531 |

2352 7590 11/25/2008

OSTROLENK FABER GERB & SOFFEN
1180 AVENUE OF THE AMERICAS
NEW YORK, NY 100368403

| EXAMINER | |
|---|---|
| DARNO, PATRICK A | |
| ART UNIT | PAPER NUMBER |
| 2169 | |
| MAIL DATE | DELIVERY MODE |
| 11/25/2008 | PAPER |

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

**BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES**

Application Number: 10/767,780
Filing Date: January 28, 2004
Appellant(s): FUKUI, SHINGO

Max Moskowitz
For Appellant

**EXAMINER'S ANSWER**

This is in response to the appeal brief received 11/06/2008 appealing from the Office action mailed 05/05/2008.

**(1) Real Party of Interest**

A statement identifying the real party of interest is contained in the brief.

**(2) Related Appeals and Interferences**

The Examiner is not aware of any related appeals, interferences, or judicial proceedings which directly affect or will be affected by or have a bearing on the Board's decision in the pending appeal.

**(3) Status of Claims**

The statement of the status of claims contained in the brief is correct.

**(4) Status of Amendments After Final**

There are no unentered amendments.

**(5) Summary of Claimed Subject Matter**

The summary of claimed subject matter contained in the brief is correct.

**(6) Grounds of Rejection to be Reviewed on Appeal**

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

**(7) Claims Appendix**

The copy of the appealed claims contained in the Appendix to the brief is correct.

**(8) Evidence Relied Upon**

- Michael M. Swift and Anne Hopkins, University of Washington and Peter Brundrett, et al., Microsoft Corporation, "Improving the Granularity of Access Control for Windows 2000", Proceedings of teh 6th ACM Symposium on Access Control Models and Technology (SACMAT 01) (Chantilly, VA., May), ACM, New York (2001)
- US 2004/0186845 Fukui [admitted prior art] **Pub:** 09-23-2004 **Filed:** 01-28-2004
- US 20030187854 Fairweather **Pub:** 10-02-2003 **Filed:** 02-03-2003

**(9) Grounds of Rejection**

The following ground(s) of rejection are applicable to the appealed claims:

***Claim Rejections - 35 USC § 102***

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

Claims 1-8, 18, 20-27, 35 and 37 are rejected under 35 U.S.C. 102(b) as being anticipated by non-patent literature article titled "Improving the Granularity of Access Control for Windows 2000" authored by Michael M. Swift et al. (hereinafter "Swift").

**Claim 1:**

Swift discloses an information sharing method for holding information owned by at least one unit user on a storage device in a tree structure provided for each unit user, said tree structure including a plurality of nodes sequentially arranged from a home root node to a plurality of leaf

nodes, such that said information corresponds to each of said nodes to manage an availability condition of each of said nodes *(Swift: page 9, section 2.2, lines 4-6 and page 9, Fig. 5 and page 15, section 3.4, lines 3-5 and page 20, Fig. 11 and page 22, Fig. 13 and page 25, section 4.5, lines 1-3; All citations from the Swift reference use the page number on the bottom of the article submitted in the Applicant's IDS. The article in its entirety is 40 pages, the first page being 1 the last being 40.)*, said method comprising:

a first step in which a computer refers to the availability condition of each of said nodes on said storage device in response to an availability condition manipulation request for changing the availability condition of one of said nodes *(Swift: page 20, section 4.2, lines 4-6, and page 20, section 4.2, lines 15-17 and page 22, section 4.3, lines 9-11 and page 22, Fig. 13)*, to determine whether or not said availability condition manipulation request can be executed while satisfying a condition that while the availability condition can be changed at multiple nodes of the tree structure, the number of times of changes in the availability condition is limited to one at maximum along any of paths from said home root node to any one of the plurality of leaf nodes *(Swift: page 11, section 2.4, lines 2-7 and page 16, lines 42-44 and page 17, section 4, lines 15-21 and page 18, lines 22-23 and page 23, lines 1-3 and page 25, section 4.5 and lines 1-10)*;

a second step in which said computer executes the availability condition manipulation request such that said condition is satisfied when the availability condition manipulation request is determined as executable in said first step *(Swift: page 11, section 2.4, lines 2-7 and page 16, lines 42-44 and page 17, section 4, lines 15-21 and page 18, lines 22-23 and page 23, lines 1-3 and page 25, section 4.5 and lines 1-10; It is clear from the references that the access rights (availability conditions) can only be made in one place in the hierarchical structure. Only access manipulations which satisfy this rule are allowed to proceed.)*, and provides a determination that the availability condition manipulation request is not executable

when the availability condition manipulation request is determined as not executable in said first step *(Swift: page 21, lines 40-44 and page 25, section 4.5, lines 1-3)*; and

a third step in which said computer refers to said availability condition in response to a tree structure manipulation request for modifying said tree structure, and executes the tree structure manipulation request such that said condition continues to be satisfied *(Swift: page 12, line 20 - page 13, line 3 and page 17, section 4, lines 15-18 and page 18, lines 4-7 and page 20, section 4.2, lines 11-12 and page 22, Fig. 13 and page 25, section 4.5, lines 1-12; It is abundantly clear that the system is restricted to a maximum of one availability condition (access control list) change. The reference shows that this condition is maintained whenever changes to the availability condition are requested or the structure is manipulated. Furthermore, it is clear from the references that the system maintains this condition in order to reduce the cost of access time because availability conditions only need to be checked in one place instead of checking availability conditions of each node along a path.)*.

**Claim 2:**

Swift discloses all the elements of claim 1, as noted above, and Swift further discloses wherein said first step includes:

when said availability condition manipulation request involves setting an availability condition *(Swift: page 11, section 2.4, lines 2-7 and page 11, section 3, lines 6-8 and page 17, section 4, lines 15-18 and page 25, section 4.5, lines 1-10)*, determining that said availability condition manipulation request is executable when the availability condition of a node under manipulation is the same as that of the home root node, or is a change start point of the availability condition in said tree structure *(Swift: page 11, section 2.4, lines 2-7 and page 16, lines 42-44 and page 17, section 4, lines 15-21 and page 18, lines 22-23 and page 23, lines 1-3 and page 25, section 4.5 and lines 1-10; All of these references reiterate*

*the fact that the availability condition can only be changed a maximum of one time. Since this is the case, at least one of these two conditions MUST be true in order for the predetermined condition of one access change to hold. Specifically, a request to change the availability condition of a node is granted (executable) if the availability condition is the same as the root because (by satisfying the predetermined condition) no change took place yet. And it's granted if it is a change start point because (again by satisfying the predetermined condition) that must be the only change since only one is allowed. )*, and determining that said availability condition manipulation request is not executable when the availability condition of said node under manipulation is different from that of said home root node, and is not said change start point *(Swift: page 11, section 2.4, lines 2-7 and page 16, lines 42-44 and page 17, section 4, lines 15-21 and page 18, lines 22-23 and page 23, lines 1-3 and page 25, section 4.5 and lines 1-10; Again, in order for this the predetermined condition to hold true, another access change cannot be granted if a node is different from the home root because that means an access change has already occurred. The same reasoning applies to the change start point. These limitations must occur if the Swift references inventive feature of one access change on a path of the hierarchy is to hold.)*.

**Claim 3:**

Swift discloses all the elements of claim 1, as noted above, and Swift further discloses wherein said first step includes:

when said availability condition manipulation request involves clearing an availability condition, determining that said availability condition manipulation request is executable when a node under manipulation is a change start point of the availability condition in said tree structure, and determining that said availability condition manipulation request is not executable when said node under manipulation is not said change start point *(Swift: page 11, section 2.4, lines 2-7 and page 16, lines 42-44 and page 17, section 4, lines 15-21 and page 18, lines 22-23 and page 23, lines 1-3 and page 25, section*

*4.5 and lines 1-10; Again, the predetermined condition of one access change on a path in the hierarchy dictate that these limitations must occur in the teachings of Swift.).*

**Claim 4:**

Swift discloses all the elements of claim 1, as noted above, and Swift further discloses wherein said first step includes:

determining that said availability condition manipulation request is not executable when a node under manipulation intended by said availability condition manipulation request is a home root node *(Swift: page 21, lines 34-38 and page 21, lines 40-44).*

**Claim 5:**

Swift discloses all the elements of claim1, as noted above, and Swift further discloses wherein said second step includes:

when said availability condition manipulation request involves setting an availability condition *(Swift: page 11, section 2.4, lines 2-7 and page 11, section 3, lines 6-8 and page 17, section 4, lines 15-18 and page 25, section 4.5, lines 1-10)*, setting the availability condition of a node under manipulation as requested by said availability condition manipulation request, and setting the same availability condition to all nodes included in a maximum partial tree in which said node under manipulation is in a position of a root *(Swift: see at least page 25, section 4.5, lines 1-10; It has been made abundantly clear that a maximum of one availability condition occurs in a hierarchy of nodes presented in the swift reference. Further, it is extremely clear that this access is propagated to all nodes under the node which was changed (maximum partial tree).).*

**Claim 6:**

Swift discloses all the elements of claim 1, as noted above, and Swift further discloses wherein said second step includes:

when said availability condition manipulation request involves clearing availability condition, clearing the availability of a node under manipulation, and setting the same availability condition as that of said node under manipulation to all nodes included in a maximum partial tree in which said node under manipulation is in position of a root *(Swift: see at least page 25, section 4.5, lines 1-10; Again, it is extremely clear that when an access control change is made (a new change or deletion/modification of a prior change), the effects of the change are propagated to all nodes under the node which was changed (maximum partial tree).)*.

**Claim 7:**

Swift discloses all the elements of claim 1, as noted above, and Swift further discloses wherein said third step includes:

when said tree structure manipulation request involves creating a new node, creating said new node at a requested location *(Swift: page 11, section 3, lines 6-8 and page 12, section 3.1, lines 20-21 and page 13, lines 1-3)*.

**Claim 8:**

Swift discloses all the elements of claim 7, as noted above, and Swift further discloses wherein said third step further includes:

setting the same availability condition of a parent node of said new node to said new node after creating said new node *(Swift: page 18, lines 4-10)*.

**Claim 18:**

Swift discloses all the elements of claim 1, as noted above, and Swift further discloses wherein each of said nodes in said tree structure is classified into at least one of an unchanged node having the same availability condition as the home root node; a change start node having an availability condition different from that of said home root node and different from that of a parent node; and a change takeover node having an availability condition different from that of said home root node and the same as that of a parent node *(Swift: page 20, section 4.2, lines 19-21; See the INHERITED_ACE flag. This is clearly indication of at least a 'change takeover node'. Based on the predetermined condition that must be satisfied according to the Swift reference, the Examiner believes the other categories of classification exist in the Swift reference as a result maintaining the availability condition.)*, said classification being added to information on said availability condition as a change state type of each of said nodes for management, wherein said computer refers to said change state type for examining said availability condition *(Swift: page 20, section 4.2, lines 19-24)*.

**Claim 20:**

Claim 20 is rejected under the same reasons set forth in the rejection of claim 1.

**Claim 21:**

Claim 21 is rejected under the same reasons set forth in the rejection of claim 2.

**Claim 22:**

Claim 22 is rejected under the same reasons set forth in the rejection of claim 3.

**Claim 23:**

Claim 23 is rejected under the same reasons set forth in the rejection of claim 4.

**Claim 24:**

Claim 24 is rejected under the same reasons set forth in the rejection of claim 5.

**Claim 25:**

Claim 25 is rejected under the same reasons set forth in the rejection of claim 6.

**Claim 26:**

Claim 26 is rejected under the same reasons set forth in the rejection of claim 7.

**Claim 27:**

Claim 27 is rejected under the same reasons set forth in the rejection of claim 8.

**Claim 35:**

Claim 35 is rejected under the same reasons set forth in the rejection of claim 18.

**Claim 37:**

Claim 37 is rejected under the same reasons set forth in the rejection of claim 1.

***Claim Rejections - 35 USC § 103***

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 9-17 and 28-34 are rejected under 35 U.S.C. 103(a) as being unpatentable over Swift and further in view of the admitted prior art of U.S. Patent Application Publication Number 2004/0186845 issued to Shingo Fukui (hereinafter "Fukui APA").

**Claim 9:**

Swift discloses all the elements of claim 1, as noted above. The Examiner is convinced that the Swift reference, at the very least, implicitly discloses all the elements of claim 9 as presented by the Applicant. However, it is noted for the record that the Swift reference does not appear to explicitly disclose wherein said third step includes:

when said tree structure manipulation request involves duplicating a node group comprising at least one node, creating a duplicate of said node group at a requested location.

However, Fukui APA discloses:

wherein said third step includes when said tree structure manipulation request involves duplicating a node group comprising at least one node, creating a duplicate of said node group at a requested location *(Fukui APA: paragraph [0033] and Fig. 8)*.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the teachings of Swift with the teachings Fukui APA at the time the invention was made. The skilled artisan would have been motivated to improve the teachings of Swift per the above in order to increase a user's control over the structure of a hierarchical access control system. Furthermore, it is noted that duplicating operations for manipulating a tree structure are admitted by the Applicant to be prior art and, therefore, are clearly well known in the art *(Fukui APA: paragraph [0033] and Fig. 8)*.

**Claim 10:**

The combination of Swift and Fukui APA discloses all the elements of claim 9, as noted above, and Swift further discloses wherein said third step further includes:

setting the same availability condition set to the parent node of a root node of said node group to said nodes which make up the duplicate of said node group after creating the duplicate of said node group *(Swift: page 18, lines 4-10)*.

**Claim 11:**

Swift discloses all the elements of claim 1, as noted above. The Examiner is convinced that the Swift reference, at the very least, implicitly discloses all the elements of claim 11 as presented by the Applicant. However, it is noted for the record that the Swift reference does not appear to explicitly disclose wherein said third step includes:

when said tree structure manipulation request involves moving a node group comprising at least one node, moving said node group to a location under a requested destination node.

However, Fukui APA discloses:

when said tree structure manipulation request involves moving a node group comprising at least one node, moving said node group to a location under a requested destination node *(Fukui APA: paragraph [0034] and Fig. 8)*.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the teachings of Swift with the teachings Fukui APA at the time the invention was made. The skilled artisan would have been motivated to improve the teachings of Swift per the above in order to increase a user's control over the structure of a hierarchical access control system. Furthermore, it is noted that moving operations for manipulating a tree structure are admitted by the Applicant to be prior art and, therefore, are clearly well known in the art *(Fukui APA: paragraph [0034] and Fig. 8)*.

**Claim 12:**

The combination of Swift and Fukui APA discloses all the elements of claim 11, as noted above, and Swift further disclose wherein said third step further includes:

performing one processing operation of a plurality of different availability condition modification processing operations depending on the availability condition of each of said nodes included in said node group after moving said node group *(Swift: see at least page 18, lines 4-10)*.

**Claim 13:**

The combination of Swift and Fukui APA discloses all the elements of claim 12, as noted above, and Swift further discloses wherein said plurality of different availability condition modification processing operations comprises processing for maintaining the availability condition of each of said nodes included in said node group *(Swift: page 18, lines 4-10 and page 25, section 4.5, lines 1-10)*, processing for setting the same availability condition of said destination node to each of said nodes *(Swift: page 18, lines 4-10)*, and processing for querying a user whether said processing for maintaining the availability condition or said processing for setting the same availability condition is performed *(Swift: page 21, lines 38-40)*.

**Claim 14:**

The combination of Swift and Fukui APA discloses all the elements of claim 11, as noted above, and Swift further discloses wherein said third step further includes:

performing one processing operation of a plurality of different availability condition modification processing operations depending on whether the availability condition of said destination node is different from that of the home root node after moving said node group *(Swift: see at least page 18, lines 4-10)*.

**Claim 15:**

The combination of Swift and Fukui APA discloses all the elements of claim 14, as noted above, and Swift further discloses wherein said plurality of different availability condition modification processing operations comprises processing for maintaining the availability condition of each of said nodes included in said node group *(Swift: page 18, lines 4-10 and page 25, section 4.5, lines 1-10)*, processing for setting the same availability condition of said destination node to each of said nodes *(Swift: page 18, lines 4-10)*, and processing for querying a user whether said processing for maintaining the availability condition or said processing for setting the same availability condition is performed *(Swift: page 21, lines 38-40)*.

**Claim 16:**

The combination of Swift and Fukui APA discloses all the elements of claim 14, as noted above, and Swift further discloses wherein said third step further includes:

performing one processing operation of a plurality of different availability condition modification processing operations depending on the availability condition of each of said nodes included in said node group after moving said node group *(Swift: see at least page 18, lines 4-10)*.

**Claim 17:**

The combination of Swift and Fukui APA discloses all the elements of claim 16, as noted above, and Swift further discloses wherein said plurality of different availability condition modification processing operations comprises processing for maintaining the availability condition of each of said nodes included in said node group *(Swift: page 18, lines 4-10 and page 25, section 4.5, lines 1-10)*, processing for setting the same availability condition of said destination node to each of said nodes *(Swift: page 18, lines 4-10)*, and processing for querying a user whether

said processing for maintaining the availability condition or said processing for setting the same availability condition is performed *(Swift: page 21, lines 38-40)*.

**Claim 28:**

Claim 28 is rejected under the same reasons set forth in the rejection of claim 9.

**Claim 29:**

Claim 29 is rejected under the same reasons set forth in the rejection of claim 10.

**Claim 30:**

Claim 30 is rejected under the same reasons set forth in the rejection of claim 11.

**Claim 31:**

Claim 31 is rejected under the same reasons set forth in the rejection of claim 12.

**Claim 32:**

Claim 32 is rejected under the same reasons set forth in the rejection of claim 13.

**Claim 33:**

Claim 33 is rejected under the same reasons set forth in the rejection of claim 14.

**Claim 34:**

Claim 34 is rejected under the same reasons set forth in the rejection of claim 15.

Claims 19 and 36 are rejected under 35 U.S.C. 103(a) as being unpatentable over Swift and further in view of U.S. Patent Application Publication Number 2003/0187854 issued to John Fairweather (hereinafter "Fairweather").

**Claim 19:**

Swift discloses all the elements of claim 1, as noted above, but Swift does not appear to explicitly disclose wherein said tree structure includes a node which is a short-cut to another node.

However, Fairweather discloses wherein said tree structure includes a node which is a short-cut to another node *(Fairweather: Fig. 1; Clearly the tree structure in Fig. 1 includes a node 120 which is a 'short-cut' to node 130.)*.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the teachings of Swift with the teachings of Fairweather noted above. The skilled artisan would have been motivated to improve the teachings of Swift per the above such that a short-cut from a given node allows a node to access another node without having to retrace steps back through the root node. Access time is therefore reduced because fewer nodes have to be traversed through the tree in order to reach the desired destination node.

**Claim 36:**

Claim 36 is rejected under the same reasons set forth in the rejection of claim 19.

**(10) Response to Arguments**

**Argument #1:**

**Appellant Argues:**

The Examiner believes that Swift discloses the limitation of Claims 1, 20, and 37 requiring that along a path from the home root node to the leaf node the availability condition changes not more than once. Applicants respectfully disagree.

Swift states:

> Type-specific inheritance and static inheritance allow centralized management by propagating changes through a hierarchy of objects, so that access control changes are made **in one place**. These features support delegation by allowing an administrator to grant access to **a single type of object**, or even a single property on a single type of object. In addition, that access is propagated both to existing objects and to new objects when they are created. Swift, page 25, third paragraph.

In other words, with reference to Fig. 5 of Swift, access control changes to each of the individual "Jane User" objects is "made in one place." However, this does not mean that in any path of the hierarchy from the home root node "Company" to any of the leaf nodes "Jane User", the number of times of changes in the availability condition is limited to one at maximum.

**Examiner Responds:**

Examiner is not persuaded. It is noted that the Examiner cited six references in Swift which disclose the Appellant's claim limitation, however, the Appellant has only argued one of the references. Each of the six references cited are now reproduced below:

> We also wanted to **allow administrators to set access control at a single point** in the Active Directory, **and let that policy flow to all appropriate objects below that point.** [Swift: page 11, section 2.4, lines 4-7]

> Another potential improvement for evaluating ACLs is to cluster the ACEs in an ACL that grants access to a particular property **to reduce the number of entries that must be inspected**. [Swift: page 16, lines 42-44]

> Administrators must therefore be able to **change permissions at one place** in the directory and **let the effects propagate down** either to all objects or only those of the appropriate type. Using the access control inheritance mechanism from Window's NT, which was designed with a file system in mind, all objects within a container inherit the same access control. **Furthermore, changes to access control at the root of a tree overwrite all changes lower in the tree**. [Swift: pg. 17, lines 15-21]

> **Only a single ACL must be evaluated** for most access checks. [Swift: pg. 18, lines 22-23]

> However, there benefit comes during an access check: **with static inheritance, only the ACL on the object itself must be inspected**. With dynamic inheritance, as shown in Figure 12, multiple ACLs must be inspected. [Swift: pg. 23, lines 1-3]

> Type-specific inheritance and static inheritance allow centralized management **by propagating changes through a hierarchy of objects, so that access control changes are only made in one place.** [Swift: page 25, lines 1-3] This approach provides the **major benefit** of dynamic inheritance, which is centralized administration but lowers the cost at access time because ACLs along the whole path do not need to be evaluated. [Swift: pg. 25, lines 8-10]

It is easily seen from the above references that "access control" changes are only made once in a hierarchy of objects. Furthermore, there is a reason that access control changes are only made once, and that is to limit the amount of nodes that need to be checked while verifying access [Swift: pg. 18, lines 22-23 and pg. 17, lines 15-21 and pg. 25, lines 8-10]. The reason Swift can limit the access control checks is because, as the references show, Swift limits the access control changes so that they only occur in one, single place [Swift: page 11, section 2.4, lines 2-7 and page 16, lines 42-44 and page 17, section 4, lines 15-21 and page 18, lines 22-23 and page 23, lines 1-3 and page 25, section 4.5 and lines 1-10].

Since it appears that each and every element of the Appellant's claimed invention is either disclosed or suggested by the prior art of record, the Appellant's claims remain rejected under the reasons set forth in the preceding office action.

**Argument #2:**

**Appellant Argues:**

Swift specifically indicates that these access control changes are different for different nodes. Swift specifically indicates that these access control changes are different for different nodes. Specifically, as Swift disclosed in reference to its Fig. 5, the "Company" node has properties that will only be accessible by the company administrators, the "Departments" node has properties that will only be accessible by company administrators and department administrators, department administrators and group administrators, and, finally, the Jane User node has properties that will only be accessible by company administrators, department administrators, group administrators, and Jane User.

Therefore, while the changes are made from "one place" in the Archive Directory, the availability condition along a particular path may change multiple times in the system of Swift.

**Examiner Responds:**

Examiner is not persuaded. The Appellant continuously argues that Fig. 5 shows **properties** which can be inherited. The Examiner does not deny that **properties** are inherited in the tree structure of Swift Fig. 5. The Examiner's rejections simply set forth that **access control** changes only occur at a **single point** from root to leaf along any given path. Swift is abundantly clear with respect to this issue [Swift: page 11, section 2.4, lines 2-7 and page 16, lines 42-44 and page 17, section 4, lines 15-21 and page 18, lines 22-23 and page 23, lines 1-3 and page 25, section 4.5 and lines 1-10]. And the reason Swift permits changes at only a **single point** along any path is to limit the amount of node access control checks that need to be performed [Swift: pg. 18, lines 22-23 and pg. 17, lines 15-21 and pg. 25, lines 8-10]. If the program has encountered an access control change, there is no need to check the access rights of each node thereafter, because the access control changes are propagated after the single access control change, to each and every existing node, or newly created node.

Since each and every element of the Appellant's claimed invention is either disclosed or suggested by the prior art of record, the Appellant's claims remain rejected under the reasons set forth in the preceding office action.

## (11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the Examiner in the Related Appeals and Interferences section of this Examiner's Answer.

**(12) Conclusion**

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Patrick Darno

/Patrick A. Darno/

Examiner

Art Unit 2169

11-23-2008

**Conferees:**

Mohammad Ali

/Mohammad Ali/

Supervisory Patent Examiner, Art Unit 2169

/Hosain T Alam/

Supervisory Patent Examiner, Art Unit 2166